Linear-Size Nonobtuse Triangulation of Polygons

Marshall Bern[1] Scott Mitchell[2] Jim Ruppert[3]

Report RNR-94-003, February 1994

NAS Systems Division
Applied Research Branch
NASA Ames Research Center, Mail Stop T27A-1
Moffett Field, CA 94035

Abstract We give an algorithm for triangulating n-vertex polygonal regions (with holes) so that no angle in the final triangulation measures more than $\pi/2$. The number of triangles in the triangulation is only $O(n)$, improving a previous bound of $O(n^2)$, and the worst-case running time is $O(n \log^2 n)$. The basic technique used in the algorithm, recursive subdivision by disks, is new and may have wider application in mesh generation. We also report on an implementation of our algorithm.

[1]Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304

[2]Applied and Numerical Mathematics Dept., Sandia National Laboratories, Albuquerque, NM 87185. This work was supported by the Applied Mathematical Sciences program, U.S. Department of Energy Research and by the U.S. Department of Energy under contract DE-AC04-76DP00789.

[3]The author is an employee of Computer Sciences Corporation. This work was supported through NASA Contract NAS 2-12961. E-mail address: ruppert@nas.nasa.gov.

Linear-size Nonobtuse Triangulation of Polygons

Marshall Bern * Scott Mitchell † Jim Ruppert ‡

February 10, 1994

Abstract

We give an algorithm for triangulating n-vertex polygonal regions (with holes) so that no angle in the final triangulation measures more than $\pi/2$. The number of triangles in the triangulation is only $O(n)$, improving a previous bound of $O(n^2)$, and the worst-case running time is $O(n \log^2 n)$. The basic technique used in the algorithm, recursive subdivision by disks, is new and may have wider application in mesh generation. We also report on an implementation of our algorithm.

1. Introduction

The triangulation of a two-dimensional polygonal region is a fundamental problem arising in computer graphics, physical simulation, and geographical information systems. Most applications demand not just any triangulation, but rather one with triangles satisfying certain shape and size criteria [9]. In order to satisfy these criteria, one typically allows triangles to use new vertices, called *Steiner points*, that are not vertices of the input polygon. The number of Steiner points should not be excessive, however, as this would increase the running time of computations.

Throughout the application areas named above, it is generally true that large angles (that is, angles close to π) are undesirable. Babuška and Aziz [2] justified this aversion for one important application area by proving convergence of the finite element method [25] as triangle sizes diminish, so long as the maximum angle is bounded away from π. They also gave an example in which convergence fails when angles grow arbitrarily flat. (An elementary example in which large angles spoil convergence is Schwarz's paradox [22].)

Any bound smaller than π implies convergence in Babuška and Aziz's model, but a bound of $\pi/2$ on the largest angle has special importance. First of all, any stricter non-varying requirement would also bound the smallest angle away from zero; for some inputs (such as a long, skinny rectangle) this forces the triangulation to contain a number of triangles dependent on the geometry—not just on the combinatorial complexity—of the input. Second, a nonobtuse triangulation is necessarily a (constrained) Delaunay triangulation [8].

*Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304

†Applied and Numerical Mathematics Dept., Sandia National Laboratories, Albuquerque, NM 87185. This work was supported by the Applied Mathematical Sciences program, U.S. Department of Energy Research and by the U.S. Department of Energy under contract DE-AC04-76DP00789.

‡NASA Ames Research Center, M/S T045-1, Moffett Field, CA 94035. The author is an employee of Computer Sciences Corporation, supported under NASA contract NAS 2-12961.

Third, a nonobtuse triangulation admits a *perpendicular planar dual*, that is, an embedding in which dual edges cross at right angles. Such an embedding is convenient for the "finite volume" method [25]. Finally, a nonobtuse triangulation has better numerical properties [3, 27]. In particular, Vavasis [27] recently proved that for finite element problems with physical characteristics that vary enormously over the domain, a nonobtuse mesh implies faster convergence of a certain numerical method.

These properties have established nonobtuse triangulation as a desirable goal in finite element mesh generation. Several heuristic methods have been developed to compute nonobtuse triangulations [4, 20]. Baker, Grosse, and Rafferty [3] gave the first provably-correct algorithm. Their algorithm also bounds the smallest angle away from zero, and hence necessarily uses a number of triangles dependent upon input geometry. See [16] for a more recent algorithm of this type.

¿From the point of view of theoretical computer science, however, it is important to determine the inherent complexity of nonobtuse triangulation, apart from no-small-angle triangulation. Bern and Eppstein [8] devised a nonobtuse triangulation algorithm using $O(n^2)$ triangles, where n is the number of vertices of the input domain. This result demonstrates a fundamental complexity separation between bounding large angles and bounding small angles. Bern, Dobkin, and Eppstein [7] later improved this bound to $O(n^{1.85})$ for convex polygons.

In this paper, we improve these bounds to linear, using an entirely different—and more widely applicable—technique. Aside from sharpening the theory, our new algorithm boasts other advantages: it parallelizes, thereby placing nonobtuse triangulation in the class $\mathcal{NC}$; and it does not use axis-parallel grids, so the output has no preferred directions. Our algorithm also improves results of Bern et al. [7] on no-large-angle triangulation. The superseded results include an algorithm guaranteeing a maximum angle of at most $5\pi/6$ that uses $O(n \log n)$ triangles for simple polygons and $O(n^{3/2})$ triangles for polygons with holes.

2. Overview of the Algorithm

Our algorithm consists of two stages. The first stage (Section 3) packs the domain with non-overlapping disks, tangent to each other and to sides of the domain. The disk packing is such that each region not covered has at most four sides (either straight sides or arcs), as shown in Figure 1(a). The algorithm then adds edges (radii) between centers of disks and points of tangency on their boundaries, thereby dividing the domain into small polygons as shown in Figure 1(b).

The second stage (Section 4) triangulates the small polygons using Steiner points located only interior to the polygons or on the domain boundary. Restricting the location of Steiner points ensures that triangulated small polygons fit together so that neighboring triangles share entire sides. Figure 1(c) shows the resulting all-right-triangle triangulation.

This algorithm is circle-based, rather than grid-based like the previous polynomial-size nonobtuse triangulation algorithm [8]. Analogously, the problem of no-small-angle triangulation has grid-based [10] and circle-based [23] solutions. In retrospect, circle-based algorithms offer a more natural way to bound angles, as well as meshes more intrinsic to the input domain. This conclusion is supported by two more examples. In the second stage of

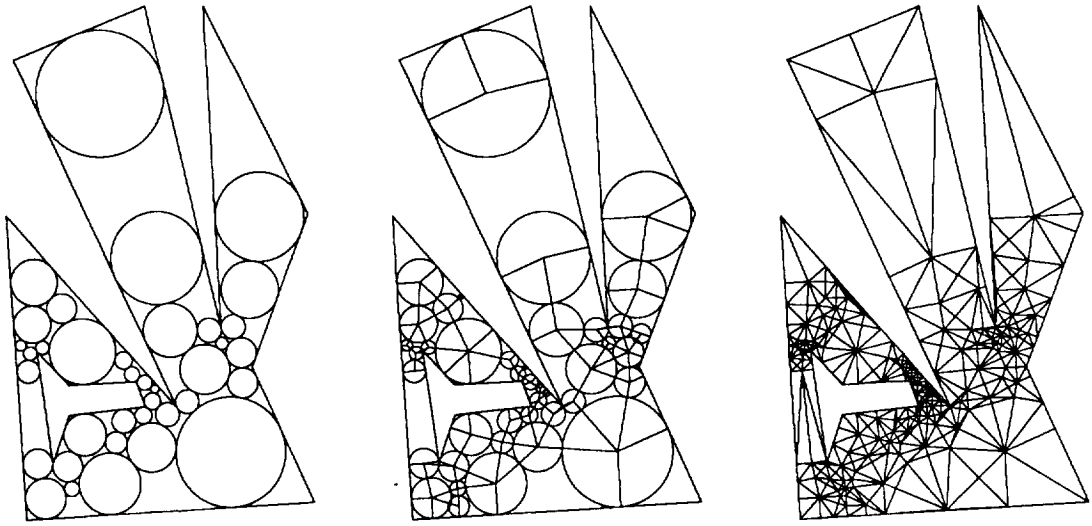

Figure 1. (a) Disk packing. (b) Induced small polygons. (c) Final triangulation.

this paper's algorithm, certain misshapen small polygons cause technical difficulties; these are neatly solved by packing in more disks. (One of these additional disks is the second from the left along the bottom side of Figure 1(b).) In other recent work, Mitchell uses the "angle buffering" property of circles to give a triangulation, restricted to use only interior Steiner points, with linear size and largest angle nearly as small as possible [18].

3. Disk Packing

In this section, we describe the first stage of the algorithm. Let P denote the input: a region of the plane bounded by a set of disjoint simple polygons with a total of n vertices. An *arc-gon* is a simple polygon with sides that are arcs of circles. The circles may have various radii, including infinity (which implies a straight side).

Throughout the disk-packing stage, we make use of the *generalized Voronoi diagram* (GVD), which is defined by proximity to both edges and vertices. The interior points of polygonal region P are divided into cells according to the nearest vertex of P, or the nearest edge (viewing each edge as an open segment). The resulting partition consists of a set of bisectors, either line segments or parabolic arcs; it is essentially the same as the *medial axis* [21]. The GVD can be similarly defined for arc-gons, or more generally for arbitrary collections of points, segments, and circular arcs. The GVD of a collection of n points, segments, and arcs can be computed in time $O(n \log n)$ [13].

The disk-packing stage consists of three smaller steps. First, one or two disks are placed at each vertex of the polygon. Second, holes in the polygon are connected to the boundary by adding disks tangent to two holes, or to a hole and the outer boundary. Third, disks are added to the as-yet-uncovered regions (called *remainder regions*), recursively reducing their complexity until all have at most four sides.

Disks at Corners. The first step preprocesses P so that we need only consider arc-gons with angle 0 at each vertex. At every convex vertex of P, we add a small disk tangent to

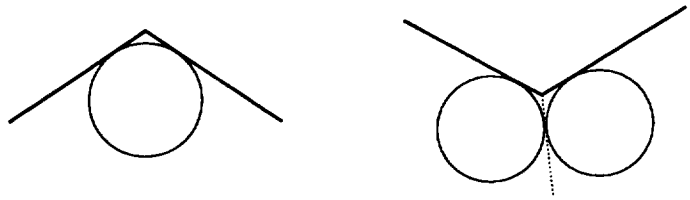

Figure 2. Adding disks at (a) convex and (b) concave corners of polygonal region P.

both edges, as shown in Figure 2(a). At every concave vertex of P, we add two disks of equal radii, tangent to the edges, and tangent to the angle bisector at the corner, as shown in Figure 2(b). We choose radii small enough that disks lie within P, and none *overlap* (that is, intersect at interior points). This step isolates a small 3- or 4-sided sided remainder region at each corner of P. The large remainder region is an arc-gon of $2n + r = O(n)$ sides, where n is the number of vertices of P and r is the number of concave corners.

The first step can be implemented in time $O(n \log n)$ using the GVD of P. By checking the adjacencies of GVD cells, we can determine the nearest non-incident edge for each vertex v of P; one-eighth this distance gives a safe radius for the disks next to v. (Our implementation actually uses some other choices of radii.)

Connecting Holes. The second step connects polygonal holes to the outer boundary by repeatedly adding a disk tangent to two or more connected components of the boundary. (At this point, a step-one disk touching a hole boundary is considered to be part of the hole.) At the end, the large remainder region is bounded by a simply-connected arc-gon with $O(n)$ sides. Every corner of this arc-gon has angle 0, since each results from a tangency.

The second step can be implemented in time $O(n \log^2 n)$. We use a data structure that answers queries of the following form: given a query point p, which data object (vertex, edge, or disk) will be hit first by an expanding circle tangent to a vertical line through p (tangent at p and to the left of the line)? Such a query can be answered using Fortune's $*$-map [13], a sort of warped Voronoi diagram.

The initial set of data objects includes the edges, vertices, and disks attached to the outer boundary of the input polygon. The first query point is the leftmost point on a hole. The answer determines a disk D entirely contained within the polygon, touching both the hole and the outer boundary. Disk D is inserted into the query data structure, along with the vertices, edges and disks of the hole. Each subsequent query is performed using the leftmost point of all remaining holes. Altogether, the queries yield a set of disks connecting all holes and the exterior of the polygon.

For a static set of data objects, the $*$-map can be built in time $O(n \log n)$ [13], and standard planar subdivision search techniques [21] yield an $O(\log n)$ query time. In our case, the set of data objects is not fixed, since edges and a disk are added following each query. A trick due to Bentley and Saxe [5] allows dynamic insertions to the query structure, with query time $O(\log^2 n)$ and amortized insert time $O(\log^2 n)$. The trick is to divide the n data objects among $O(\log n)$ data structures, one for each bit in the binary representation of n. A query searches all data structures in $O(\log^2 n)$ time. An insertion rebuilds all the data structures corresponding to bits that change. The total time required for n insertions is $O(n \log^2 n)$.

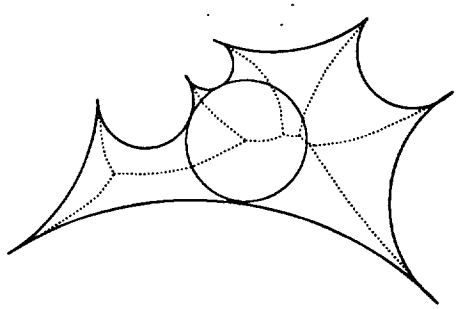

Figure 3. A disk tangent to three edges of an arc-gon is centered at a vertex of the GVD.

Reducing to 3- and 4-Sided Remainder Regions. As shown in Figure 3, a disk tangent to three sides of an arc-gon must be centered at a vertex of the GVD. Such a disk divides the region enclosed by the arc-gon into four pieces: the disk itself and three smaller regions bounded by arc-gons. The final step of the disk-packing stage adds a linear number of disks and reduces all remainder regions to ones bounded by 3- and 4-sided arc-gons. After the first two steps, the only remainder region with more than four sides is simply connected; hence the edges of its GVD form a tree.

Let A be a simply connected n-vertex arc-gon. To subdivide A, we add a disk tangent to three sides, not all of which are consecutive. If three consecutive sides were used, no progress would be made: the three resulting arc-gons would have 3, 3, and n sides. Non-consecutive sides guarantee that each resulting arc-gon will have at most $n-1$ sides.

Lemma 1. *It is possible to reduce all remainder regions to at most 4 sides, by packing $O(n)$ non-overlapping disks into arc-gon A.*

Proof: Each vertex of the GVD corresponds to a disk tangent to three sides of A. If A has at least five sides, then there is a vertex v of the GVD that is adjacent to two non-leaf vertices of the GVD; a disk centered at v is tangent to three sides of A that are not all consecutive.

Now let $d(n)$ be the maximum number of disks need to reduce an n-sided arc-gon to 3- and 4-sided remainder regions. We prove $d(n) \leq n-4$ by induction on n. The base cases are $d(3) = 0$ and $d(4) = 0$.

For the inductive step, notice that adding one disk produces three new arc-gons. (We can simply ignore extra tangencies in the degenerate case of four or more tangencies.) Suppose the new arc-gons have k, l, m sides, respectively, with $3 \leq k \leq l \leq m$. Since we are choosing non-consecutive sides, as guaranteed by Lemma 1, $m < n$. Counting 1 for the added disk, we have that $d(n) \leq 1 + d(k) + d(l) + d(m)$. Since the disk divides three sides, and is itself divided in three places, we have $k + l + m = n + 6$.

First suppose $k = 3$. Since we are choosing non-consecutive sides, $l \geq 4$, so

$$\begin{aligned} d(n) &\leq 1 + d(3) + d(l) + d(m) \\ &\leq 1 + 0 + (l-4) + (m-4) \\ &= (l+m) - 7 = (n+3) - 7 = n-4. \end{aligned}$$

When $k \geq 4$, we have $d(n) \leq 1 + d(k) + d(l) + d(m)$. By induction, $d(n) \leq 1 + (k-4) + (l-4) + (m-4)$, which is equal to $(k+l+m) - 11 = (n+6) - 11 = n-5$. ∎

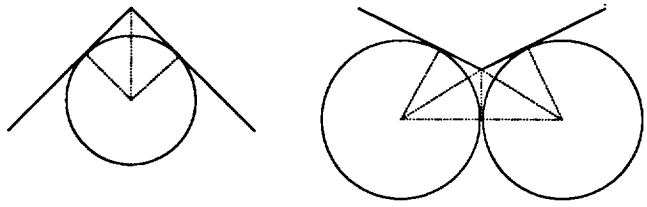

Figure 4. Triangulating regions with vertices of the polygonal region P.

Finally we comment on running time. Any tree contains a vertex, called a *centroid*, whose removal leaves subtrees of size at most two-thirds the original size. By choosing a disk centered at a centroid of the GVD of A, we split A into arc-gons A_1, A_2, and A_3. We imagine splitting A_1, A_2, and A_3 in parallel, so that altogether there will be $O(\log n)$ splitting stages, each involving a set of arc-gons of total complexity $O(n)$. If we recompute GVD's from scratch after each splitting stage, we obtain total time $O(n \log^2 n)$. This can be improved to $O(n \log n)$ by rebuilding GVD's in linear time, using an adaptation of [1].

4. Triangulating the Pieces

We now describe the second stage of our algorithm. At this point, polygonal region P has been partitioned into disks and remainder regions with three or four sides, either straight or circular arcs. Each circular arc of a remainder region R is naturally associated with a pie-shaped sector, namely the convex hull of the arc and the center of the circle containing the arc. We denote the union of R and its associated sectors by R^+. These *augmented* remainder regions define a decomposition of P into simple polygons with disjoint interiors.

In this section, we show how to triangulate each R^+ region. All Steiner points will lie either on straight sides of R (that is, along P's boundary) or interior to R^+. Thus we never place Steiner points on the radii bounding sectors, and triangulated R^+ regions will fit together at the end. Our triangulation method is given in three cases: remainder regions with vertices of P, three-sided remainder regions, and four-sided remainder regions. The first two cases are easy, but the last is quite intricate. In all cases, triangulating a single R^+ region takes $O(1)$ time, so altogether the running time of the second stage is $O(n)$.

Remainder Regions with Vertices of P. Every vertex of P was isolated by one or two disks in the first step of the algorithm. The resulting regions R^+ can be triangulated with at most four right triangles, as shown in Figure 4, by adding edges from the disk centers to the points of tangency and the vertex of P.

Three-Sided Remainder Regions. A three-sided remainder region R without a vertex of P is bounded by three circular arcs, so that arcs meet tangent at the vertices of R. Here we are considering a straight side to be an arc of an infinitely large circle. We call a Steiner point in an augmented remainder region R^+ *safe* if it lies either interior to R^+ or on the boundary of P.

Lemma 2. *If R is a three-sided remainder region, then R^+ can be triangulated with at most six right triangles, adding only safe Steiner points.*

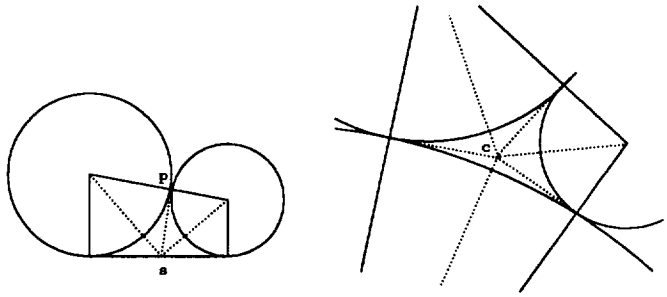

Figure 5. Three-sided remainder regions: (a) with a straight side, (b) with only finite-radius arcs.

Proof: First assume that R has a straight side (necessarily at most one), and view R so that this straight side forms a horizontal base. The augmented region R^+ is a trapezoid with two vertical sides, and a subdivision point p along its slanted top side. We cut perpendicularly from p (that is, tangent to both arcs) across R until we hit the base, and there add a safe Steiner point s. We add edges from s to the centers of the arcs' circles to divide R^+ into four right triangles, as shown in Figure 5(a).

Now assume all the sides of R are arcs of finite radius. Notice that R^+ is a triangle with subdivided sides. Moreover, the subdivision points along the sides of R^+ are exactly the tangency points of the inscribed circle of R^+. (This follows from the fact that the inscribed circle makes each corner of R^+ incident to two edges of equal length.) So we add the circle's center c and edges from c to all the vertices around R^+, dividing R^+ into six right triangles, as shown in Figure 5(b). ∎

Four-Sided Remainder Regions. A four-sided remainder region R is bounded by four circular arcs (possibly of infinite radius) that meet tangent at the vertices of R. Lemma 3 states two interesting properties of these regions.

Lemma 3. *The arcs of R have total measure 2π. The vertices of R are cocircular.*

Proof: If all arcs have finite radius, then the sum of the measures of the arcs of R is identical to the sum of the measures of the angles at the corners of R^+. For straight sides, we imagine further augmenting R with "infinite sectors" of angle 0.

Next we show that the vertices are cocircular. Let C_1 and C_3 be finite-radius circles containing opposite arcs of R. (Here notice that if R has two straight sides, they must be opposite.) Assume the two lines that are externally tangent to both C_1 and C_3 meet at a point x. There exists an inversive transformation ([11], pp. 77–95) of the projective plane that maps x to infinity and hence the two external tangent lines to parallel lines. The transformed circles C_1' and C_3', corresponding to C_1 and C_3, have equal size, so the vertices of the transformed remainder region R' form an isosceles trapezoid. It is easy to see that any isosceles trapezoid has cocircular vertices. The inverse of the original inversive transformation maps the circle containing the vertices of R' to a circle containing the vertices of R. ∎

Now if we are lucky, the region R^+ can be triangulated with 16 right triangles, as shown in Figure 6 Here we have added the center c of the circle through R's vertices in

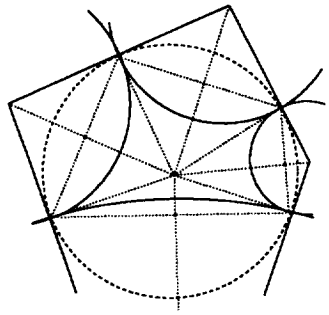

Figure 6. The good case for four-sided remainder regions.

order to form four *kites* (quadrilaterals with two adjacent pairs of equal-length sides). This triangulation, however, can fail in two different ways: (1) if one of the arcs of R measures more than π (a *reflex* arc), then the angles at the corresponding vertex of R^+ will measure more than $\pi/2$; and (2) if center c lies outside the convex hull of R, then it lies on the wrong side of one of the chords and will introduce unwanted intersections. Each of these difficulties will be handled by adding yet another disk.

First assume R has a reflex arc on circle C_3. Add another disk C^*, tangent to C_3 and C_1 (the circle containing the arc opposite C_3), such that the center of C^* lies on the line joining the centers of C_1 and C_3. The new disk C^*—unlike any of the disks used up until this point—may overlap an old disk and produce a self-intersecting remainder region, as shown in Figure 7. Lemma 3 still holds for self-intersecting remainder regions. Region R^+, formed as before by adding the associated sectors to R, remains a simple polygon with subdivision points on its sides, specifically a triangle with three subdivisions on one side and one on each of the others. The next lemma shows how to triangulate R^+ with a generalization of the method of Lemma 2.

Lemma 4. *Let R be a self-intersecting four-sided remainder region resulting from breaking up a reflex four-sided remainder region by the addition of C^*. Then R^+ can be triangulated with at most 12 right triangles, adding only safe Steiner points.*

Proof: We may assume that all arcs of R have finite radius. If R has a straight edge, we can apply the triangulation to a region with an infinite sector attached to the straight edge and then simply remove the resulting infinite strips.

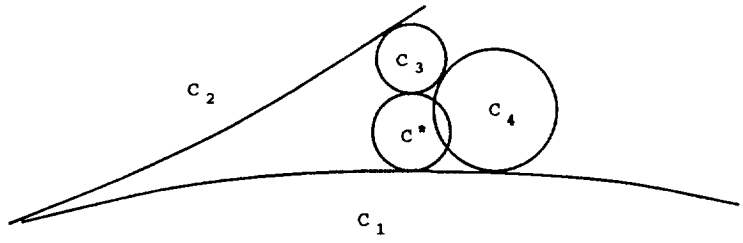

Figure 7. Adding a new circle C^* to break up a reflex remainder region.

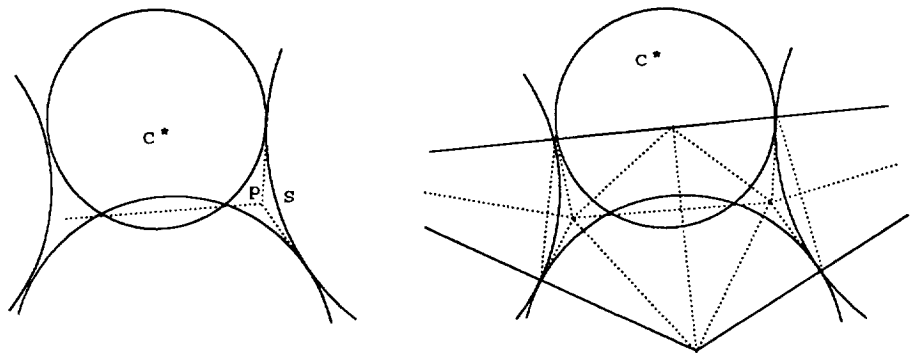

Figure 8. (a) Mutual tangents and mutual chord meet at a point. (b) Triangulation.

Consider one of the arcs S next to C^*. We claim that the lines tangent to S at its endpoints and the mutual chord of C^* and its opposite arc all meet at a single point p interior to R, as shown in Figure 8(a). This claim allows the triangulation shown in Figure 8(b).

Why is the claim true? For each of the three disks—C^*, the opposite disk, and the one with arc S—we define a *power* function. The power function of a circle with center (x_c, y_c) and radius r is $P(x,y) = (x - x_c)^2 + (y - y_c)^2 - r^2$. The power functions of two tangent circles are equal along their mutual tangent line; the power functions of two overlapping circles are equal along a line containing their mutual chord. The point p of the claim is the point at which all three power functions are equal. ∎

We now consider the second difficulty. Call a four-sided remainder region R *centered* if the convex hull of R contains the center c of the circle through R's vertices, and *uncentered* otherwise. Let the arc of R with the longest chord lie along circle C_1, and denote the other circles by C_2, C_3, and C_4, clockwise around R. (Circles through infinity handle the case of straight sides.) Assume that the line through the centers of C_1 and C_3 is vertical as in Figure 9. If R is uncentered, then c must lie below the chord on C_1.

Let t_{12} be the vertex of R at which C_1 and C_2 meet, and similarly define t_{23}, t_{34}, and t_{41}. For a disk C^* tangent to both C_1 and C_3, let S_L $(= S_L(C^*))$ be the circular arc with endpoints t_{12} and t_{23} that passes through the points at which C^* meets C_1 and C_3. Lemma 3 guarantees that such an arc exists. Similarly define S_r. Let c_L and c_r be the centers of the circles containing S_L and c_r, respectively.

Lemma 5. *There exists a disk C_c^* tangent to C_1 and C_3, such that c_L lies in the convex hull of the four points of tangency around S_L and c_r lies in the convex hull of the four points of tangency around S_r.*

Proof: First let C^* be the disk that is tangent to C_1 and C_3 such that the center of C^* lies on the line through the centers of C_1 and C_3.

Centers c_L and c_r lie on a horizontal line through the center of C^*, hence outside C_1 and C_3. But the requirements of the lemma may be violated, because c_r may lie outside the chord $t_{34}t_{41}$ of S_r (if S_r has measure less than π) or c_L may lie outside the chord $t_{12}t_{23}$ of S_L (if S_L has measure less than π). We assert that both of these bad conditions cannot occur

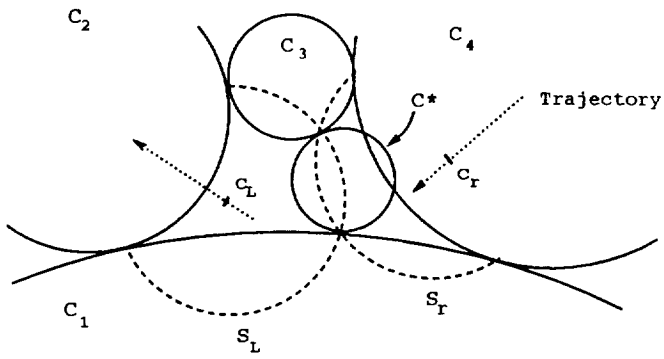

Figure 9. The trajectories of centers c_L and c_r as C^* sweeps.

at the same time. Why? It suffices to show that the sum of the measures of S_L and S_r is at least 2π. Angle $\angle t_{23}t_{*3}t_{34}$, where t_{*3} is the point of tangency of C^* and C_3, measures at least $\pi/2$, because the arc of R on C_3 measures at most π. And $\angle t_{12}t_{*3}t_{41}$ measures more than $\pi/2$, because the center of the circle through the vertices of R lies below $t_{12}t_{41}$. Hence the remaining angles at t_{*3} (those subtended by points on S_L and S_r) sum to less than π.

If neither bad condition occurs, then C^* satisfies the conditions of the lemma, and we are done. But if one of the bad conditions does occur, then we sweep C^* in the direction that could cure the condition, while keeping C^* tangent to both C_1 and C_3. If c_r lies outside $t_{34}t_{41}$, then we sweep C^* to the left in Figure 9; the other case is symmetrical.

During the leftward sweep, c_r moves towards C_1 along the perpendicular bisector of $t_{34}t_{41}$ and c_L moves towards C_2 along the perpendicular bisector of $t_{12}t_{23}$, as shown in Figure 9. These bisectors never intersect C_3, so c_L and c_r can never lie outside their chords on C_3. The chords of S_L and S_r between tangency points on C^* are never the longest chords on these arcs, so c_L and c_r also lie safely inside these chords throughout the sweep.

As c_r moves, it must pass pass through $t_{34}t_{41}$ and become good, before it reaches C_1 and becomes bad. By the arc-measure argument above, c_r must cross inside $t_{34}t_{41}$ before c_L crosses outside $t_{12}t_{23}$. Hence at some point in the sweep, both c_r and c_L satisfy the conditions of the lemma, and the C^* at this point is C_c^*. ∎

Lemma 5 breaks up uncentered, non-reflex remainder regions, but unless C_c^* coincides with the initial C^* in the sweep, adding C_c^* creates a new reflex remainder region. The following lemma finesses this final difficulty (shall we say circularity?) by triangulating both new augmented regions at once.

Lemma 6. *Let R be a non-reflex, uncentered, four-sided remainder region. Then R^+ can be triangulated into at most 28 right triangles, adding only safe Steiner points.*

Proof: Again we may assume that all arcs of R have finite radius, as a solution to this case implies a triangulation for the case of straight sides.

We start by adding the "centering" disk C_c^*, guaranteed by Lemma 5. As above, we denote the tangent point of C_c^* and C_1 by t_{*1} and the tangent point of C_c^* and C_3 by t_{*3}. In addition to t_{*1} and t_{*3}, we add the following Steiner points: the centers c_ℓ and c_r associated with arcs S_ℓ and S_r, and the midpoint m of segment $t_{*1}t_{*3}$. See Figure 10.

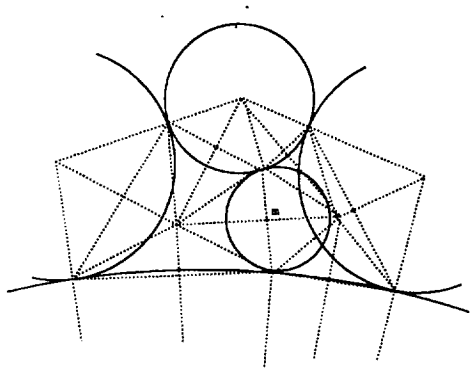

Figure 10. Triangulating R^+ when R is uncentered.

We triangulate by adding: all chords around S_ℓ and S_r; lines from c_ℓ to point m and to the centers of C_1, C_2, and C_3; and lines from c_r to point m and to the centers of C_3, C_4, and C_1. Finally we add an edge between the center of C_1 and t_{*1} and between the center of C_3 and t_{*3}.

Resulting triangles come in sets of four, each set triangulating a kite by adding its diagonals. Hence all triangles are right. (Notice that C_c^* is treated somewhat differently than the other circles: we do not use its center. Nevertheless the four triangles around m form a kite, because $t_{*1}t_{*3}$ is the mutual chord of C_c^*, S_2, and S_4.) ∎

We have now completed the proof of our main theorem.

Theorem 1. *An n-vertex polygonal region can be triangulated with $O(n)$ right triangles, in time $O(n \log n)$ for simple polygons and $O(n \log^2 n)$ for polygons with holes.* ∎

5. Implementation

We implemented our algorithm within the Matlab environment [15]. The implementation differs somewhat from the algorithm described in the text. We use several heuristics for disk placement so as to reduce the number of triangles. Also we do not bother to compute generalized Voronoi diagrams. Rather we use a simple $O(hn)$ method to connect h holes to the boundary, and we choose arbitrary disks touching three non-consecutive sides, rather than disks centered at GVD centroids. To keep the user entertained during the worst-case $O(n^2)$ running time, we display color-coded disks and triangles as they are added.

Experiments with a variety of polygonal regions show that an n-vertex input typically produces about $20n$ triangles. (The maximum observed was about $25n$ for an input with $n-3$ reflex corners.) Since a floating point representation entails roundoff, some of the right angles present in the nonobtuse triangulation become slightly obtuse. The worst test case had an angle of about $\frac{\pi}{2} + 10^{-11}$ radians (Matlab retains 16 digits), so the implementation is fairly robust, which is somewhat surprising given that our implementation often places very small disks next to very large ones.

6. Parallelizing the Algorithm

We now sketch the first $\mathcal{NC}$ algorithm for nonobtuse triangulation. We give a straightforward though rather inefficient algorithm, with parallel time $O(\log^3 n)$ and processor requirement $O(n^2)$. Both time and processors should be improvable. One bottleneck subproblem is the computation of the GVD of circular arcs; see [14] for the GVD of line segments.

Theorem 2. *An n-vertex polygonal region P (with holes) can be triangulated with $O(n)$ right triangles in $O(\log^3 n)$ time on $O(n^2)$ EREW PRAM processors.*

Proof: Using $O(n^2)$ processors—one for each vertex-edge pair—and time $O(\log n)$, we can compute the nearest non-incident edge for each vertex and hence choose appropriate radii for disks to pack into corners. The second step, connecting holes, is trickier. We first compute a minimum spanning tree (MST) of P's holes; by this we mean the shortest set of line segments S, each segment with both endpoints on the boundary of P, such that the union of S and the exterior of P is a connected subset of the plane. Using $O(n^2)$ processors and time $O(\log n)$, we compute for each vertex the nearest edge lying on a different connected component of P's boundary. We use this information to compute distances between connected components, and add to S the shortest component-joining line segment incident to each component. This reduces the number of components by at least a factor of two, so $O(\log n)$ such merging steps suffices to complete the computation of S.

Now it is not hard to show that no point of the plane is covered by more than $O(1)$ diameter disks of segments in S. Hence there is a pairwise-disjoint set of diameter disks of cardinality a constant fraction of $|S|$ [26]. It is not hard to find these disks in parallel time $O(\log n)$ using separators. We repeat the process of computing the MST (of the new connected components, holes plus disks) and finding a large independent set of diameter disks. After $O(\log n)$ cycles—for total time of $O(\log^3 n)$—we have reduced to a simply-connected arc-gon.

The third step of the disk-packing stage uses the generalized Voronoi diagram in order to find centroid disks. Using $O(n^2)$ processors and time $O(\log^2 n)$, we can compute the GVD of a set of n circular arcs as follows. We compute the equal-distance curve (bisector) for each pair of arcs. Then for each arc a, we compute the piecewise-polynomial boundary of a's cell recursively by dividing the set of bisectors into two equal halves and then merging the boundaries for each half. Two piecewise-polynomial boundaries of $O(n)$ pieces can be merged in time $O(\log n)$ on n processors. Once, the GVD has been computed, a centroid can be found in time $O(\log n)$ by alternately removing leaves and merging degree-2 paths.

Recall that the algorithm requires a "decomposition tree" of centroid disks of height $O(\log n)$, so by simply recomputing the GVD after each centroid, we obtain an overall time for the third disk-packing step of $O(\log^3 n)$. Finally, the triangulation stage consists entirely of local operations, so it is trivially parallelized. ∎

7. Conclusion

We have presented a new algorithm for nonobtuse triangulation of polygonal regions with holes. The number of triangles produced is linear in the number of vertices of the input,

a significant improvement over previous methods. This is of course worst-case optimal, resolving the question of the theoretical complexity of nonobtuse triangulation of polygons.

One direction for further work is extending the algorithm to inputs more general than polygons with holes; these inputs occur in modeling domains made of more than one material. Currently, there is an algorithm for refining a triangulated simple polygon into a nonobtuse triangulation with $O(n^4)$ triangles, and also an $\Omega(n^2)$ lower bound [8]. There is still no algorithm for polynomial-size nonobtuse triangulation of planar straight-line graphs; a solution to this problem would give another solution to "conforming Delaunay triangulation" [12]. Mitchell [17] recently showed how to triangulate planar straight-line graphs with maximum angle at most $7\pi/8$, using at most $O(n^2 \log n)$ triangles.

Another important direction is exploring whether our ideas can be used for related mesh-generation problems. For instance, disk-packing may yield a simpler algorithm for the problem of no-small-angle, nonobtuse triangulation [3, 16]. Perhaps we can use our methods to produce nonobtuse meshes with skinny triangles aligned with the boundary. (See [19] for aligned no-large-angle meshes.) Or perhaps our methods can be allied with a heuristic method called "bubble systems" [24].

Finally, higher dimensions are still a mystery. Do 3-d polyhedra admit polynomial-size triangulations without obtuse dihedral angles? Algorithms for point sets are known [6, 10].

Acknowledgments

We would like to thank Paul Chew and Shang-Hua Teng for some valuable discussions and Dan Asimov and Micha Sharir for proofs of Lemma 3.

References

[1] A. Aggarwal, L.J. Guibas, J. Saxe, and P.W. Shor. A linear time algorithm for computing the Voronoi diagram of a convex polygon. *Disc. and Comp. Geometry* 4 (1989), 591–604.

[2] I. Babuška and A. Aziz. On the angle condition in the finite element method. *SIAM J. Numer. Analysis* 13 (1976), 214–227.

[3] B.S. Baker, E. Grosse, and C.S. Rafferty. Nonobtuse triangulation of polygons. *Discrete and Comp. Geom.* 3 (1988), 147–168.

[4] R.E. Bank. *PLTMG User's Guide.* SIAM, 1990.

[5] J.L. Bentley and J.B. Saxe. Decomposable searching problems: 1. Static-to-dynamic transformation. *J. Algorithms* 1 (1980), 301–358.

[6] M. Bern, L.P. Chew, D. Eppstein, and J. Ruppert. Dihedral Bounds for Mesh Generation in High Dimensions. Manuscript, 1993.

[7] M. Bern, D. Dobkin, and D. Eppstein. Triangulating polygons without large angles. *Proc. 8th Annual ACM Symp. Computational Geometry*, 1992. To appear in *IJCGA*.

[8] M. Bern and D. Eppstein. Polynomial-size nonobtuse triangulation of polygons. *Int. J. Comp. Geometry and Applications* 2 (1992), 241–255.

[9] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. Xerox PARC Tech. Report CSL-92-1. Also in *Computing in Euclidean Geometry*, World Scientific, Singapore, 1992.

[10] M. Bern, D. Eppstein, and J.R. Gilbert. Provably good mesh generation. *Proc. 31st IEEE Symp. on Foundations of Computer Science*, 1990, 231–241. To appear in *J. Comp. System Science*.

[11] H.S.M. Coxeter. *Introduction to Geometry*. John Wiley & Sons, 1961.

[12] H. Edelsbrunner and T.S. Tan. An upper bound for conforming Delaunay triangulations. *Proc. 8th Annual ACM Symp. Computational Geometry*, 1992.

[13] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica* 2 (1987), 153–174.

[14] M.T. Goodrich, C. ÓDúnlaing, and C. Yap. Computing the Voronoi diagram of a set of line segments in parallel. *Algorithmica* 9 (1993), 128–141.

[15] *MATLAB Reference Guide*, The MathWorks, Inc., Natick, Massachusetts, 1992.

[16] E. Melissaratos and D. Souvaine. Coping with inconsistencies: a new approach to produce quality triangulations of polygonal domains with holes. *Proc. 8th Annual ACM Symp. Computational Geometry*, 1992, 202–211.

[17] S.A. Mitchell. Refining a triangulation of a planar straight-line graph to eliminate large angles. *Proc. 34th Symp. on Foundations of Computer Science*, 1993, 583–591.

[18] S.A. Mitchell. Finding a covering triangulation whose maximum angle is provably small. To appear in *Proc. 17th Annual Computer Science Conference, ACSC-17*, New Zealand, 1994.

[19] J.-D. Müller. Proven angular bounds and stretched triangulations with the frontal Delaunay method. *Proc. 11th AIAA Comp. Fluid Dynamics*, Orlando, 1993.

[20] S. Müller, K. Kells, and W. Fichtner. Automatic rectangle-based adaptive mesh generation without obtuse angles. *IEEE Trans. Computer-Aided Design* 10 (1992), 855–863.

[21] F. Preparata and M. Shamos. *Computational Geometry - an Introduction*. Springer-Verlag, 1985.

[22] J.F. Randolph. *Calculus and Analytic Geometry*. Wadsworth, 1961, 373–374.

[23] J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms*, 1993, 83–92.

[24] K. Shimada and D.C. Gossard, Computational methods for physically-based FE mesh generation. *Proc. IFIP TC5/WG5.3 8th Int. Conference on PROLAMAT*, Tokyo, 1992.

[25] G. Strang and G.J. Fix. *An Analysis of the Finite Element Method*. Prentice Hall, 1973.

[26] S.-H. Teng. *Points, Spheres, and Separators: a unified geometric approach to graph partitioning*. Ph.D. Thesis, Carnegie Mellon University, CMU-CS-91-184, 1991.

[27] S.A. Vavasis. Stable finite elements for problems with wild coefficients. Tech. Report TR93-1364, Dept. of Computer Science, Cornell University, 1993.